Java Enterprise Edition

Gabriele Tolomei

DAIS – Università Ca' Foscari Venezia

Programma del Corso

- 09/01 – Introduzione
- 10/01 – Java Servlets
- 16-17/01 – JavaServer Pages (JSP)
- 23-24/01 – Lab: Applicazione “AffableBean”
- 30-31/01 – Enterprise JavaBeans (EJB) + Lab

EJB 3.0: Introduzione

- Componenti server-side che implementano la "business logic" di un'applicazione
- Gli EJBs cooperano all'interno di un server Java EE
- I possibili "client" degli EJBs sono:
  - Componenti del Web Tier (locali o remoti al server)
  - Client remoti (ad es. Java RMI)
  - Client di Web Service (HTTP/SOAP)
- L'idea degli EJBs è quella di spostare tutta la logica applicativa al di fuori del livello "Web", in un layer appositamente dedicato

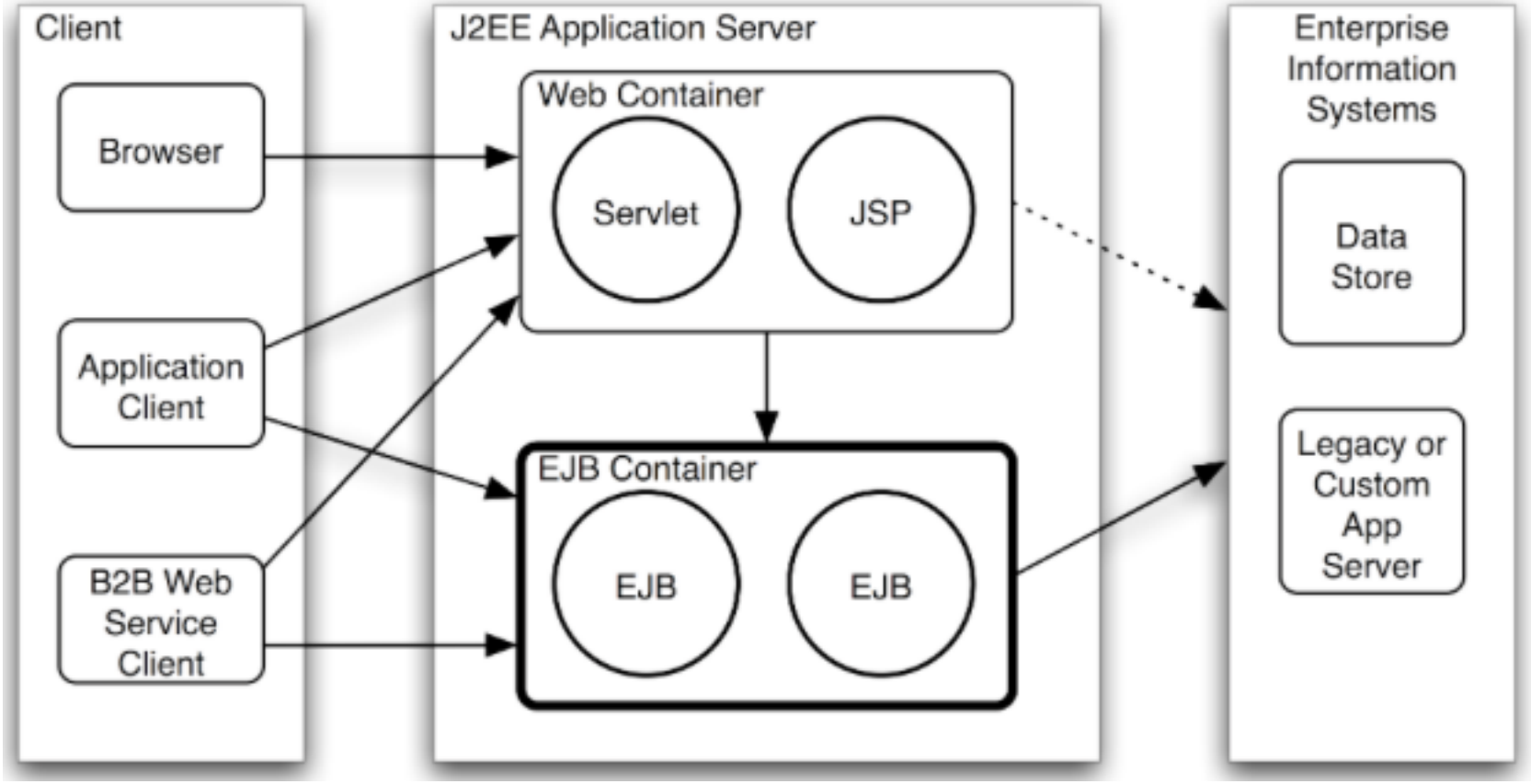
EJB 3.0: Componenti

- 2 componenti principali definiti dallo standard:
  - Session Beans → eseguono la logica applicativa, gestiscono le transazioni e il controllo dell'accesso
  - Message-Driven Beans → eseguono azioni (in modo asincrono) in risposta ad eventi (ad es. ricezione di un messaggio JMS)
- Ciascuno dei due tipi di componenti ha un proprio ciclo di vita specifico
- Il comportamento di ogni componente EJB è specificato tramite l'uso di metadati:
  - Annotazioni sul codice
  - Descrittori xml → come web.xml nel caso delle Servlet

EJB 3.0: Componenti (Nota)

- Prima di EJB 3.0 venivano considerati componenti EJB anche i cosiddetti Entity Beans
  - Poiché anch'essi erano gestiti dal container EJB
- Gli Entity Beans rappresentano le tabelle di un DB relazionale
- A partire da EJB 3.0 gli Entity Beans non sono più gestiti dal container EJB ma usano servizi implementati da un'opportuna interfaccia (JPA)

Il Container EJB



Il Container EJB

- Gli EJBs vengono eseguiti all'interno di un apposito container
- Uno degli ambienti di esecuzione fornito dalle implementazioni di server Java EE
- Fornisce servizi a livello di "sistema":
  - Transazioni (compreso quelle distribuite)
  - Sicurezza
  - Persistenza (tramite JPA = Java Persistance API)
- Esattamente come i componenti web sono eseguiti nel Web container (Servlet/JSP container), gli EJBs necessitano dei servizi forniti da un EJB container

EJB: Vantaggi

- Semplificano lo sviluppo di applicazioni enterprise distribuite di grandi dimensioni
  - Gli sviluppatori di EJBs devono focalizzarsi solo sulla logica e lasciare il resto al container EJB
  - Lo sviluppo della UI è semplificato vista la separazione tra interfaccia e logica
  - Portabilità di EJBs su altri server Java EE (a patto che questi implementino almeno le stesse specifiche)

EJB: Quando?

- Tecnologia utile nelle seguenti situazioni:
  - Applicazioni con una logica complessa che deve essere “scalabile” (ad es. distribuzione del carico su diverse istanze server)
  - Applicazioni che richiedano transazioni (distribuite) che garantiscano l’integrità dei dati (il container si occupa di gestire un corretto accesso concorrente ad oggetti condivisi)
  - Applicazioni accedute da diversi tipi di client

EJB: Svantaggi

- Fino alla specifica EJB 2.1 le applicazioni EJB erano difficilmente sviluppabili, deployabili e testabili
  - Molti file descriptors per la configurazione
  - Unit testing automatico impossibile
  - …
- Alcuni svantaggi persistono anche in EJB 3.0
  - Ad es., l'uso di EJB vincola e costringe la scelta di un server Java EE "completo" (Tomcat, che è solo un Web container, non può essere usato!)

EJB: Session Beans

- Componenti riusabili che contengono la logica applicativa
- I client interagiscono con i Session Beans localmente o remotamente
- L'accesso avviene tramite invocazione di metodi pubblici dell'EJB
- 2 tipi di Session Beans:
  - Stateless Session Beans
  - Stateful Session Beans

EJB: Stateless Session Beans

- Esegue un task per conto di un client di cui NON mantiene lo stato
  - O meglio lo mantiene per la sola durata dell'invocazione del metodo
- Termina al ritorno della chiamata del metodo
- Non mantiene informazioni in memoria secondaria (disco)
- Possono essere "condivisi" e sono gli EJB più comuni
- Esempi: invio di email, conversione della valuta, …

EJB: Stateful Session Beans

- Mantiene le informazioni di stato del client attraverso chiamate multiple ai metodi dell'EJB
- Rilascia lo stato una volta che il client lo richiede o il bean "termina"
- Può dover mantenere lo stato su memoria secondaria
- Solitamente rappresenta entità temporanee come il "carrello della spesa"

EJB: Message Driven Beans

- EJB che processano i messaggi dell'applicazione in modalità asincrona
- Implementati come listener JMS
- Simili a Stateless Session Beans
  - Non mantengono informazioni di stato del client
- Un unico MDB può servire i messaggi provenienti da più client
- Gli MDB non espongono interfacce come i Session Beans

Entity Beans

- Rappresentano le tabelle di un RDBMS
- Ciascuna istanza corrisponde ad un record di una specifica tabella
- Ciascuna entità ha un identificativo univoco
- Ovviamente, le entità possono avere relazioni tra di esse
- Ricorda: a partire da EJB 3.0 gli Entity Beans sono interamente gestiti da JPA

Entity Beans

- Le classi che rappresentano Entity Beans devono rispettare i seguenti requisiti:
  - Devono essere annotate con javax.persistence.Entity
- Devono avere un costruttore vuoto
  - Senza parametri
- Non possono essere dichiarate final
- Le variabili di istanza persistenti devono essere dichiarate private e le proprietà accedibili tramite getters/setters

Session Beans: Interfacce

- A parte gli MDB i Session Beans devono definire le interfacce con cui possono essere invocati dai client
- Le interfacce consentono di isolare l'implementazione sottostante (facilitando eventuali modifiche future)
- 3 possibili interfacce:
  - Remote Interface
  - Local Interface
  - Web Service Interface

Session Beans: Remote Interface

- I client possono essere in esecuzione su JVM diverse rispetto agli EJB
- Utilizzata da:
  - componenti web (ad es. Servlet)
  - altri EJB
  - client dell'applicazione
- "Posizione" dell'EJB trasparente rispetto al client
- Il Bean può essere distribuito → scalabilità

Session Beans: Local Interface

- I client devono essere in esecuzione sulla stessa JVM rispetto a quella degli EJB
- Utilizzata da:
  - componenti web (ad es. Servlet)
  - altri EJB
- "Posizione" dell'EJB deve essere quella del client
- Accoppiamento "forte" tra client e EJB

Session Beans: Web Service Interface

- Solo per i Session Beans “Stateless”
- I client sono in esecuzione su una JVM diversa da quella degli EJB
- Utilizzata da client di Web service tramite i protocolli previsti dalle specifiche (SOAP/WSDL su HTTP)
- “Posizione” del Bean trasparente
- I client possono essere implementati in qualsiasi linguaggio di programmazione

Stateless Session Beans: Ciclo di Vita

- Istanze create dal container EJB
- Mantenute in un “pool” di istanze “pronte”
- A fronte di una chiamata di un metodo da parte di un client:
  - Il container EJB assegna alla chiamata una delle istanze pronte del bean
  - Una volta eseguito il metodo l’istanza torna nel pool

Stateful Session Beans: Ciclo di Vita

- Il client avvia una sessione
- Il costruttore di default del bean viene invocato
- Le risorse vengono “iniettate” (se presenti)
- Il metodo del bean annotato con l’etichetta @PostConstruct viene eseguito
- Da questo momento il bean rimane nella cache per eseguire le altre richieste del client

Message-Driven Beans: Ciclo di Vita

- Il bean riceve un messaggio
- Il container EJB ricerca un'istanza del bean disponibile nel pool
- Se disponibile, l'istanza viene usata
- Una volta completata l'esecuzione del metodo onMessage() l'istanza torna nel pool
- Simile al Stateless Session Bean

Java Persistence API (JPA)

- Consente di memorizzare in modo automatico dati contenuti in oggetti Java su DB relazionali
- Object-Relational Mapping (ORM)
- Le applicazioni possono gestire le tabelle dei DB relazionali come “normali” oggetti Java
- Le classi Java (entità) corrispondono 1:1 alle tabelle relazionali definite nel DB

JPA: Vantaggi

- Ha una propria sintassi SQL-like per query statiche e dinamiche
  - Java Persistence Query Language (JPQL)
  - Portabilità rispetto a vari DB
- Evita allo sviluppatore di scrivere query di "basso livello" JDBC/SQL
- Fornisce in modo trasparente servizi di caching e ottimizzazione delle performance

JPA: Provider

- Java definisce solo le specifiche standard della JPA API, così come per Servlet, JSP ed EJB
- Tali specifiche saranno poi implementate da un cosiddetto provider
  - Esattamente come Apache Tomcat fornisce un'implementazione delle specifiche Servlet/JSP (ovvero un web container)
  - Esattamente come JBoss AS fornisce un'implementazione conforme alle specifiche Java EE "complete" (web+ejb container)
- Esistono vari provider JPA:
  - EclipseLink
  - Hibernate
  - …

JPA + Eclipse

- Per aggiungere JPA al progetto EJB Eclipse
- Click dx sul progetto EJB
  - Properties → Project Facets
  → spuntare Java Persistence (JPA 1.0) → OK
- Click dx sul progetto EJB
  - New → JPA Entity from Tables…